Find the maximum of the secret function

```
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)
```

Given a website to provide datasets (https://adaphetnodes.shinyapps.io/design_of_experiments/?user_e7268) From this website we are expected to give at least one experiment with 11 values. Each values divided by commas and each experiment divided by newline. We are allowed to create 10000 datasets at max. We expected to find a maximum value of secret function that takes 11 values.

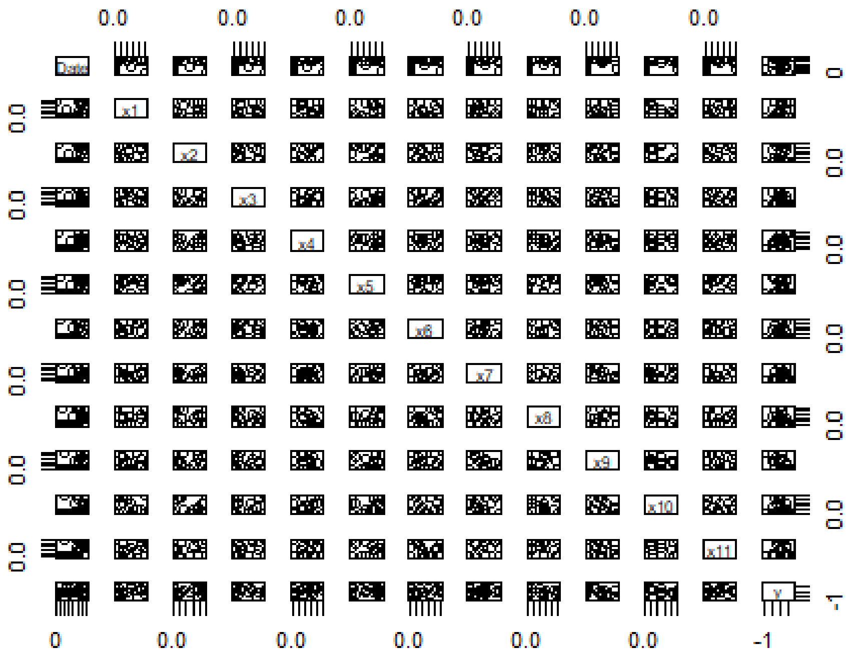
In here we are reading the data file from the csv and get a view of the value that containted for each variable.

```
d1 <- read.csv("20240118_0128_user_e7268-DoEShinnyApplication.csv")
summary(d1)

##      Date                 x1                x2                x3

##  Length:36          Min.   :0.0000   Min.   :0.0000
Min.   :0.0000
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st
Qu.:0.0000
##  Mode  :character   Median :0.0000   Median :0.0000
Median :0.0000
##                     Mean   :0.2044   Mean   :0.2533
Mean   :0.2247
##                     3rd Qu.:0.3925   3rd Qu.:0.5350   3rd
Qu.:0.3125
##                     Max.   :1.0000   Max.   :1.0000
Max.   :1.0000
##        x4               x5                x6                x7
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000   Median :0.0000   Median :0.0000
```

```
##  Mean   :0.245   Mean   :0.2833   Mean   :0.2514   Mean   :0.2269
##  3rd Qu.:0.520   3rd Qu.:0.7025   3rd Qu.:0.4400   3rd Qu.:0.4700
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       x8                x9               x10               x11

##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000

##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000

##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000

##  Mean   :0.2178   Mean   :0.1817   Mean   :0.2219   Mean   :0.2358

##  3rd Qu.:0.4300   3rd Qu.:0.3550   3rd Qu.:0.3275   3rd Qu.:0.4500

##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000

##       y
##  Min.   :-0.9873
##  1st Qu.: 1.0124
##  Median : 1.0153
##  Mean   : 1.0196
##  3rd Qu.: 1.1263
##  Max.   : 2.6621
```

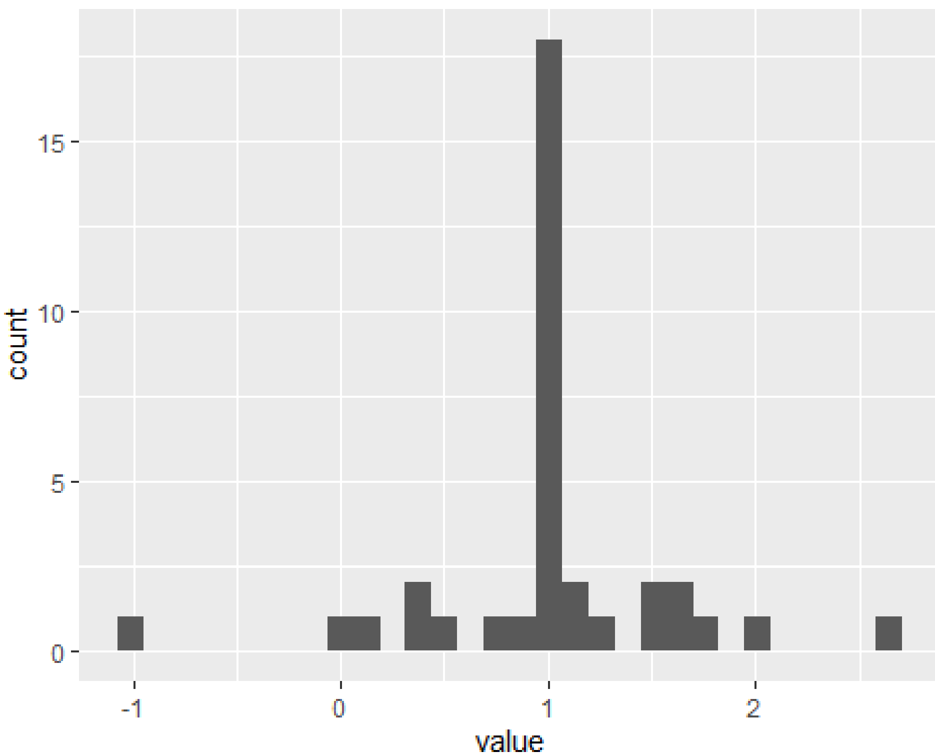
Now let’s try to see the value in graph, maybe we can find the connection or interaction between variable.

```
plot(d1)
```



I cannot see something clearly from this plot. Let's focus to the result that we have

```
d1 %>% select(-Date) %>% gather() %>% filter(key == "y") %>%
ggplot(aes(x=value, group=key)) + geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



So far, i can see the value of Y is normally around 1 but there a result that give 2 or more.

We try to analyze the impact of each value

```
res <- lm(data=d1, y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
x10 + x11)
summary(res)

##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11, data = d1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.88668 -0.14543 -0.00794  0.17174  0.75667
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.022123   0.095358  10.719 1.25e-10 ***
## x1           0.219865   0.273610   0.804  0.42953
## x2          -0.096045   0.260627  -0.369  0.71572
## x3          -0.168693   0.287162  -0.587  0.56239
## x4           1.008351   0.245968   4.100  0.00041 ***
## x5           0.231421   0.289895   0.798  0.43253
## x6           0.200538   0.288496   0.695  0.49366
## x7          -0.105928   0.313739  -0.338  0.73858
## x8          -0.006927   0.276728  -0.025  0.98024
```

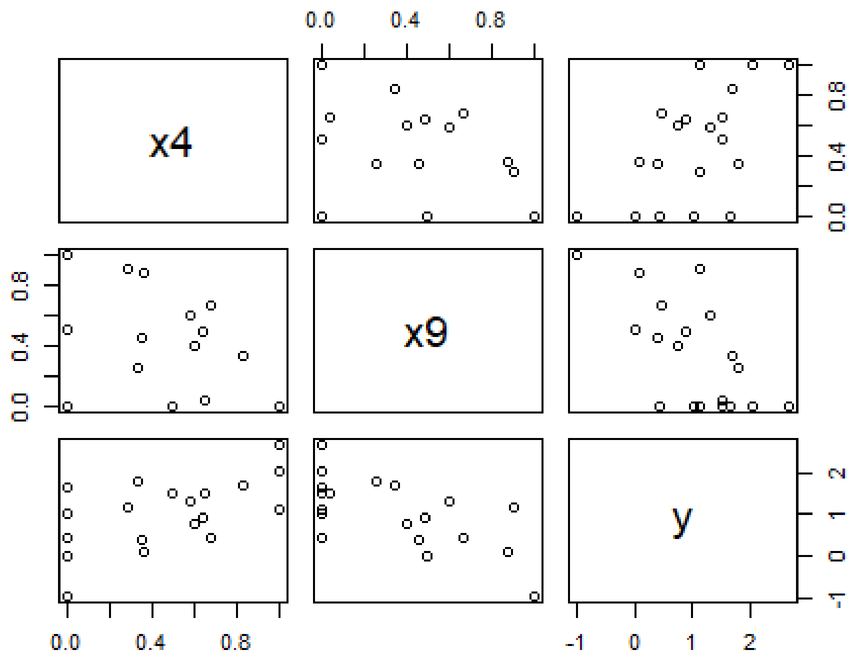
```
## x9          -1.424478   0.257757  -5.526 1.10e-05 ***
## x10         -0.419114   0.279474  -1.500  0.14675
## x11          0.123199   0.280093   0.440  0.66398
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4084 on 24 degrees of freedom
## Multiple R-squared:  0.6858, Adjusted R-squared:  0.5418
## F-statistic: 4.762 on 11 and 24 DF,  p-value: 0.0006824
```

```
anova(res)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x1         1 0.4480  0.4480  2.6853  0.114323
## x2         1 0.0032  0.0032  0.0194  0.890388
## x3         1 0.2156  0.2156  1.2926  0.266788
## x4         1 2.3203  2.3203 13.9087  0.001041 **
## x5         1 0.1382  0.1382  0.8286  0.371723
## x6         1 0.0255  0.0255  0.1529  0.699237
## x7         1 0.0249  0.0249  0.1492  0.702705
## x8         1 0.1686  0.1686  1.0105  0.324817
## x9         1 5.0115  5.0115 30.0412 1.236e-05 ***
## x10        1 0.3507  0.3507  2.1022  0.160032
## x11        1 0.0323  0.0323  0.1935  0.663980
## Residuals 24 4.0037  0.1668
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From here I can see the value of x4 and x9 is really significant.

```
d1 %>% select(-Date) -> df
plot(df[,c(4,9,12)])
```



From the graph above, we can assume that if we want to get a higher value of y, we should put x4 closer to 1 and x9 closer to 0.

Just for another test case, let’s try to create more data set. Since the previous one, the data that we use are more to 0 and 1 and just a few of them are between of them. Therefore, I create a C program to generate the random data.

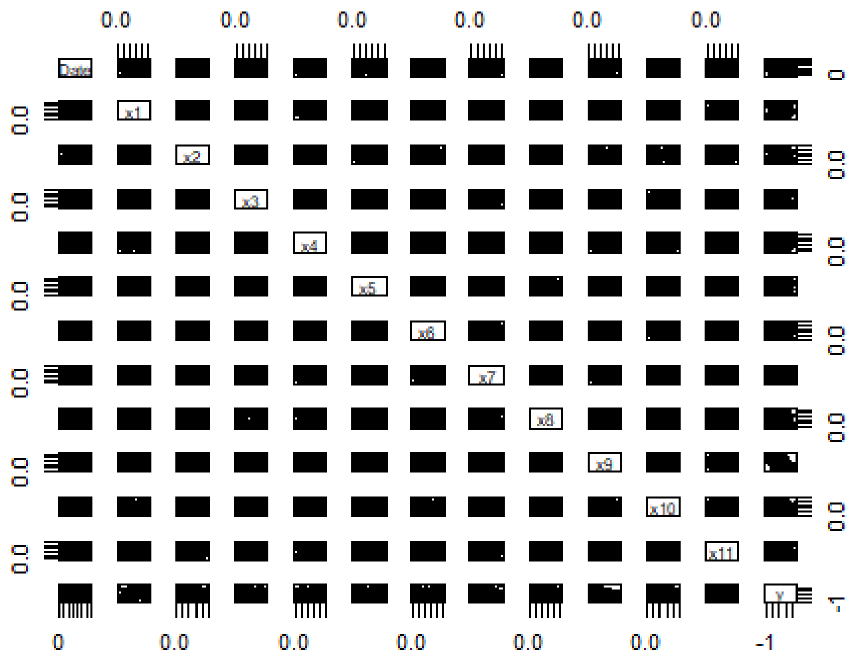
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int cr=100;
    int row = 11;
    for(int i=0;i<cr;i++){
        for(int j=0;j<row; j++){
            float value = (rand()%100)/100.0;
            printf("%.2f",value);
            if(j+1< row) printf(",");
        }
        puts("");
    }

    return 0;
}
```

Note: the code is not able to run in Rstudio (obviously), so I just run in with the C compiler. So now, let's try the new data set and let's try to do the same thing as previously.

```
d2 <- read.csv("20240201_0024_user_e7268-DoEShinnyApplication.csv")
summary(d2)

##      Date                 x1               x2               x3

##  Length:136         Min.   :0.0000   Min.   :0.0000
Min.   :0.0000
##  Class :character   1st Qu.:0.1675   1st Qu.:0.1475   1st
Qu.:0.1250
##  Mode  :character   Median :0.4050   Median :0.4300
Median :0.4600
##                     Mean   :0.4484   Mean   :0.4311
Mean   :0.4409
##                     3rd Qu.:0.7400   3rd Qu.:0.6825   3rd
Qu.:0.7125
##                     Max.   :1.0000   Max.   :1.0000
Max.   :1.0000
##       x4               x5               x6               x7

##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000

##  1st Qu.:0.2075   1st Qu.:0.1650   1st Qu.:0.1100   1st Qu.:0.1200

##  Median :0.5100   Median :0.4050   Median :0.4150   Median :0.4200

##  Mean   :0.4785   Mean   :0.4375   Mean   :0.4403   Mean   :0.4151

##  3rd Qu.:0.7425   3rd Qu.:0.7325   3rd Qu.:0.7300   3rd Qu.:0.7000

##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000

##       x8               x9              x10              x11

##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000

##  1st Qu.:0.1275   1st Qu.:0.0575   1st Qu.:0.0675   1st Qu.:0.1075

##  Median :0.3700   Median :0.3950   Median :0.3000   Median :0.4250

##  Mean   :0.4205   Mean   :0.4143   Mean   :0.3890   Mean   :0.4489

##  3rd Qu.:0.7050   3rd Qu.:0.6975   3rd Qu.:0.6775   3rd Qu.:0.7525

##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000

##       y
```

```
##   Min.   :-0.9873
##   1st Qu.: 0.4142
##   Median : 1.0140
##   Mean   : 0.9997
##   3rd Qu.: 1.4392
##   Max.   : 3.4274
```

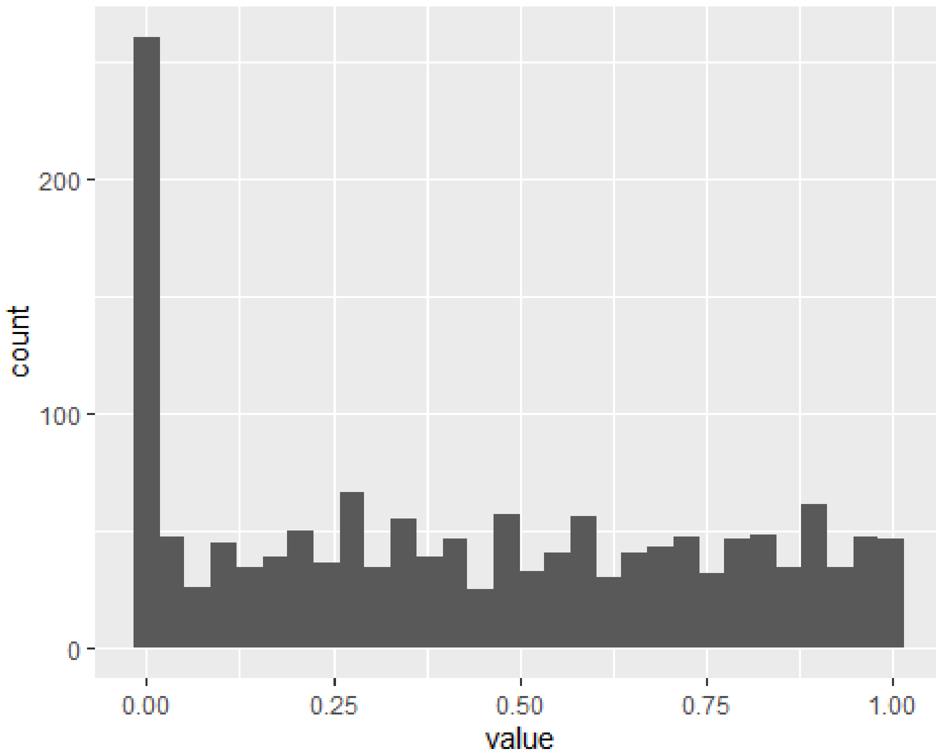
```r
plot(d2)
```



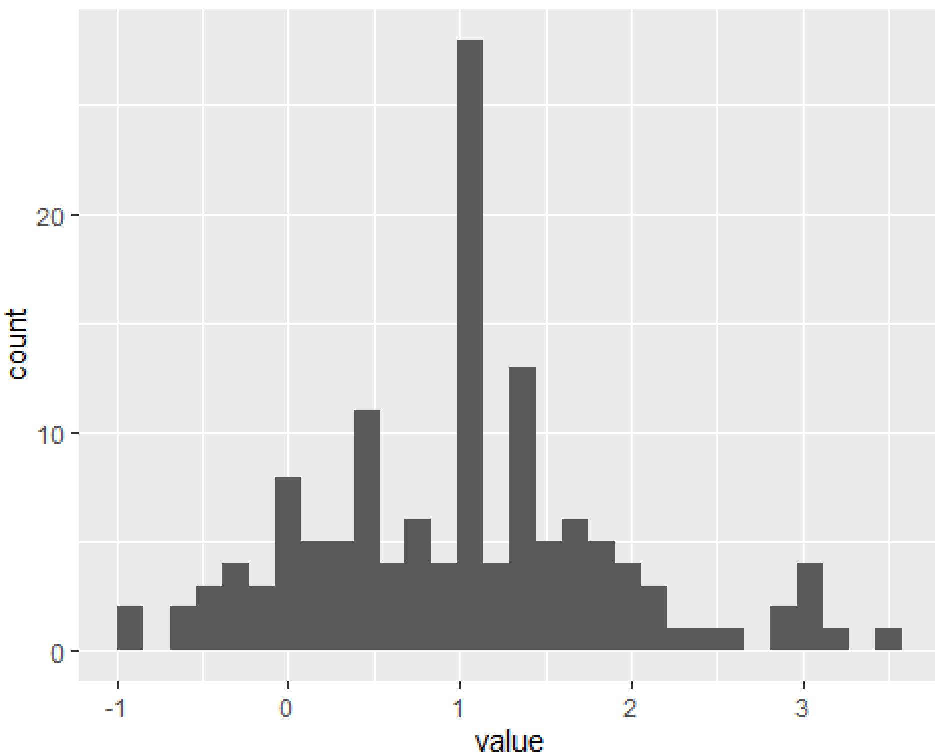
well now it even harder to see from this plot. Let's try focus to the result and variable at seperate plot

```r
d2 %>% select(-Date) %>% gather() %>% filter(key != "y") %>%
ggplot(aes(x=value, group=key)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
d2 %>% select(-Date) %>% gather() %>% filter(key == "y") %>%
ggplot(aes(x=value, group=key)) + geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Now as we can see the data is more well distributed and we found out that we might miss something since now we are able to get value higher than 3.

let's try to analyze the impact of each value linearly again

```
res <- lm(data=d2, y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
x10 + x11)
summary(res)

##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11, data = d2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46102 -0.24319 -0.02738  0.34488  1.09080
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.08214    0.12261   8.826 8.51e-15 ***
## x1           0.45974    0.16817   2.734  0.00718 **
## x2          -0.27364    0.19666  -1.391  0.16658
## x3          -0.14307    0.17699  -0.808  0.42043
## x4           0.94953    0.17513   5.422 2.96e-07 ***
## x5           0.59834    0.18649   3.208  0.00170 **
## x6           0.04238    0.17668   0.240  0.81082
## x7          -0.13249    0.17414  -0.761  0.44822
```

```
## x8           0.12910    0.16935   0.762  0.44730    
## x9          -1.89328    0.16489 -11.482  < 2e-16 ***
## x10         -0.41174    0.17238  -2.389  0.01842 *  
## x11          0.22930    0.18383   1.247  0.21462    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.5737 on 124 degrees of freedom
## Multiple R-squared:  0.615,  Adjusted R-squared:  0.5808 
## F-statistic: 18.01 on 11 and 124 DF,  p-value: < 2.2e-16
```

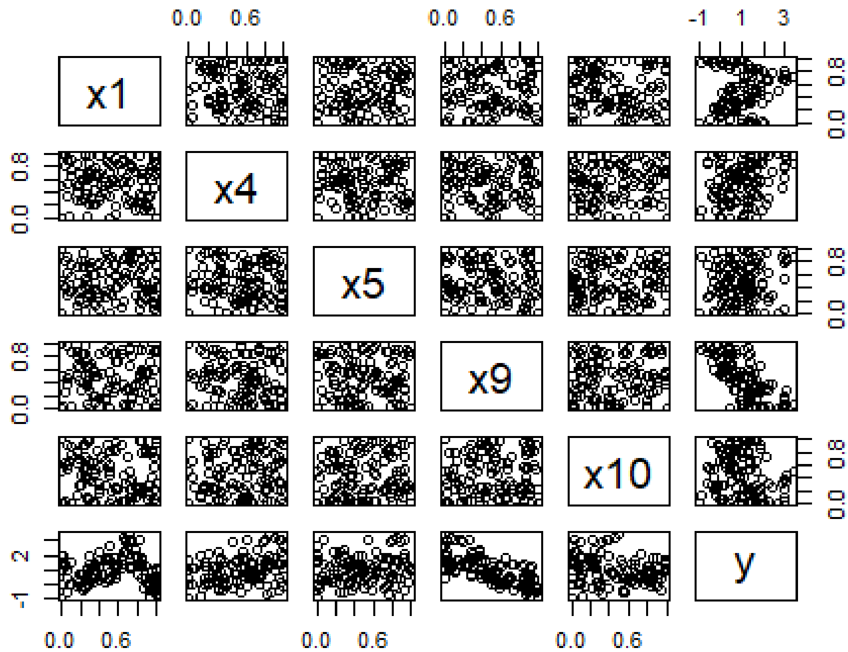
```
anova(res)
```

```
## Analysis of Variance Table
## 
## Response: y
##            Df Sum Sq Mean Sq  F value    Pr(>F)    
## x1          1  1.719   1.719   5.2246   0.02397 *  
## x2          1  0.639   0.639   1.9421   0.16593    
## x3          1  1.530   1.530   4.6506   0.03297 *  
## x4          1  8.242   8.242  25.0457 1.872e-06 ***
## x5          1  1.217   1.217   3.6965   0.05682 .  
## x6          1  0.053   0.053   0.1620   0.68797    
## x7          1  1.979   1.979   6.0144   0.01558 *  
## x8          1  0.000   0.000   0.0004   0.98434    
## x9          1 47.737  47.737 145.0554 < 2.2e-16 ***
## x10         1  1.555   1.555   4.7261   0.03161 *  
## x11         1  0.512   0.512   1.5559   0.21462    
## Residuals 124 40.808   0.329                       
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we found out that x4, x9 are really significant, x1, x5 are significant and x10 is a bit significant.

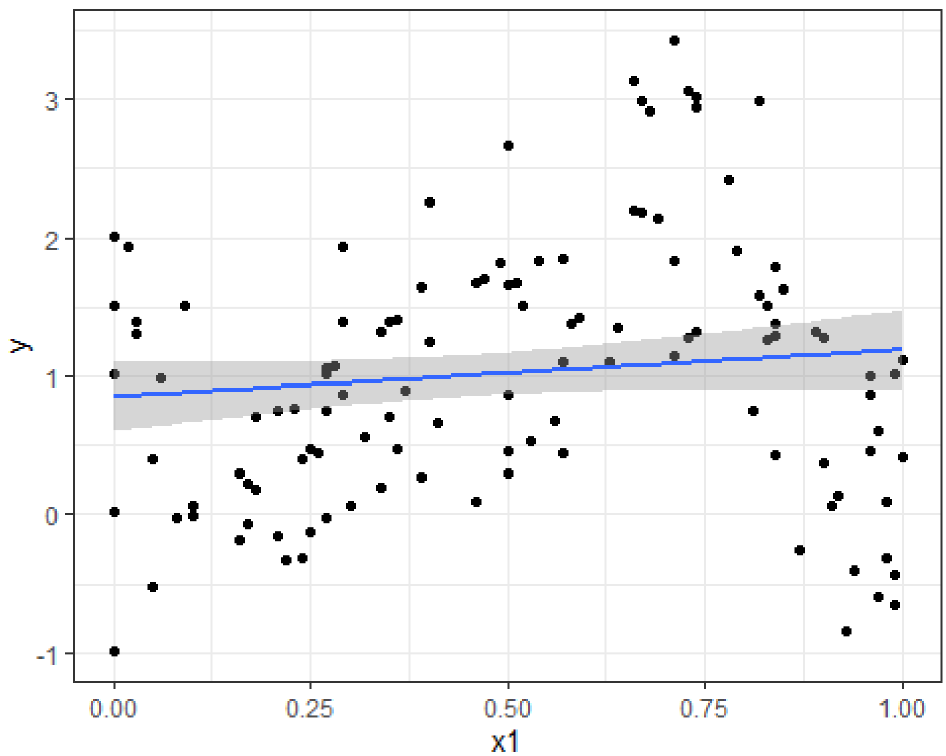
Let’s try to focus again with their value and the distribution.

```
d2 %>% select(-Date) -> df
plot(df[,c(1,4,5,9,10,12)])
```

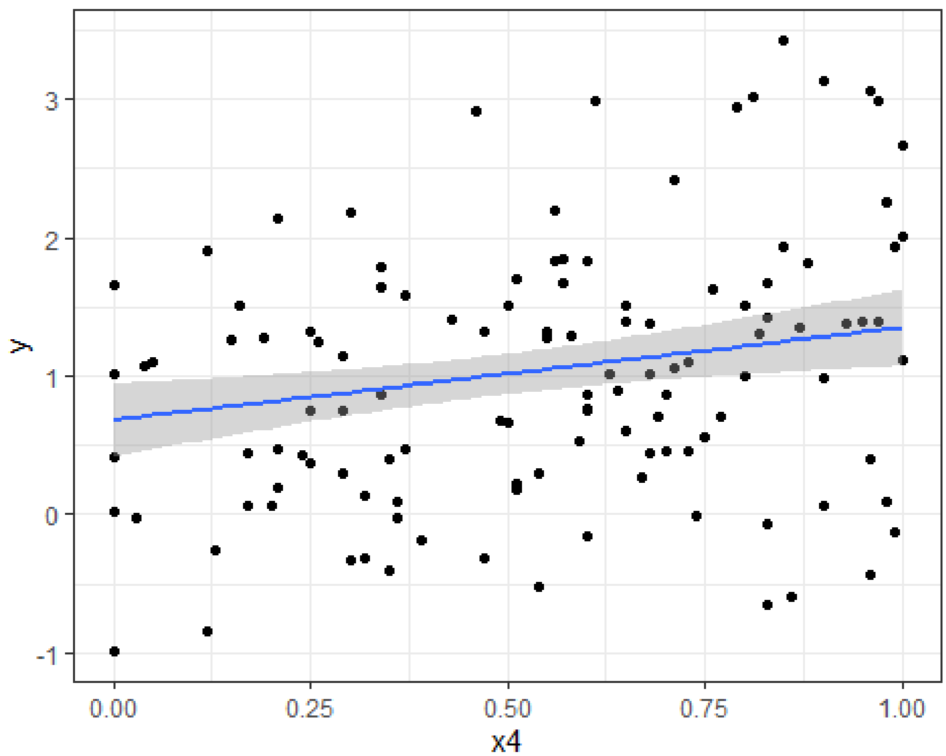


From what I
able to see for now, I can assume that to get a high y, we need to have x1 around 0.7, x4 around 0.8, x5 around 0.9, x9 around 0.1, and x10 around 0.5 for now.

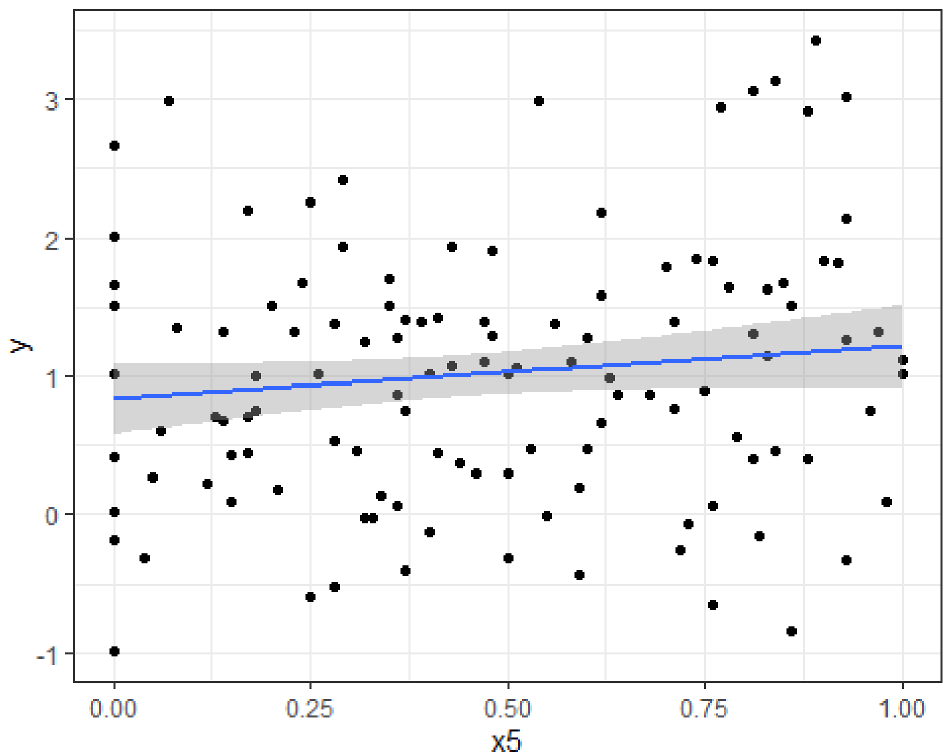
```
ggplot(d2, aes(y=y, x=x1)) + geom_point() + geom_smooth(method="lm") +
theme_bw()

## `geom_smooth()` using formula = 'y ~ x'
```



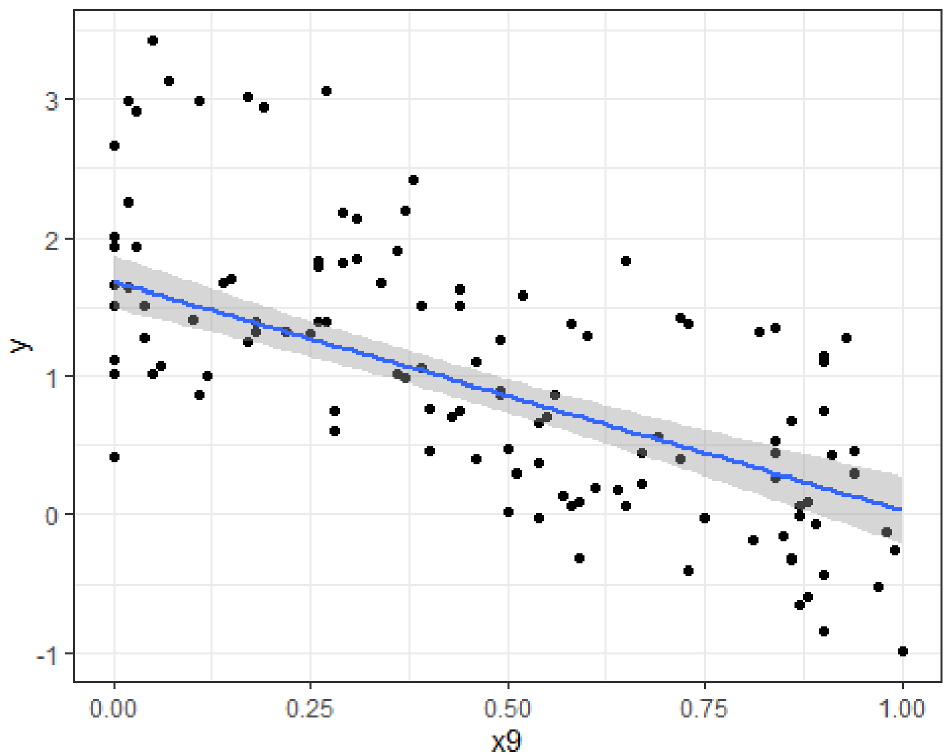

```
ggplot(d2, aes(y=y, x=x4)) + geom_point() + geom_smooth(method="lm") +
theme_bw()

## `geom_smooth()` using formula = 'y ~ x'
```



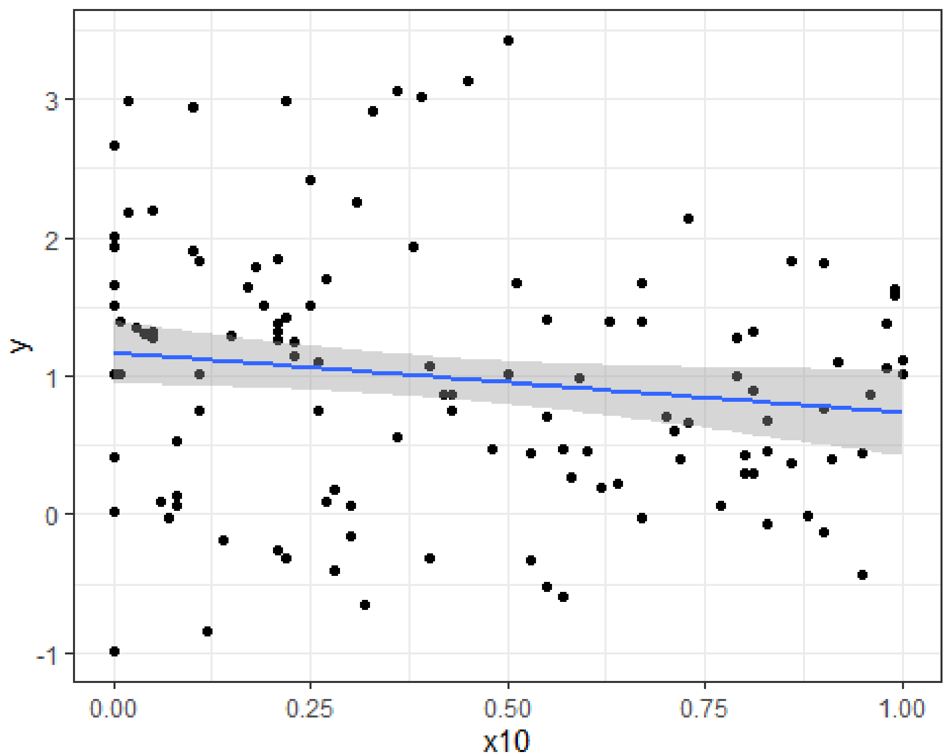

```r
ggplot(d2, aes(y=y, x=x5)) + geom_point() + geom_smooth(method="lm") +
theme_bw()

## `geom_smooth()` using formula = 'y ~ x'
```




```r
ggplot(d2, aes(y=y, x=x9)) + geom_point() + geom_smooth(method="lm") +
theme_bw()

## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(d2, aes(y=y, x=x10)) + geom_point() + geom_smooth(method="lm")
+ theme_bw()

## `geom_smooth()` using formula = 'y ~ x'
```



From the graph above, we can see that x1 is not looks like a linear one, and for x4 and x5 looks like the give the impact to y in a good way (higher their value = higher y value) even though from the plot it didn't look affect to much. Meanwhile, for x9 and x10 they give bad impact for y (the higher their value = lower y value) especially x9.

Let's try guess a new formula, especially for x1.

```
res <- lm(data=d2, y ~ poly(x1,3) + x4 + x5 + x9 + x10)
summary(res)

##
## Call:
## lm(formula = y ~ poly(x1, 3) + x4 + x5 + x9 + x10, data = d2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42125 -0.12573 -0.04598  0.15126  0.38711
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.29045    0.04451  28.991  < 2e-16 ***
## poly(x1, 3)1    1.94861    0.21489   9.068  1.8e-15 ***
## poly(x1, 3)2   -3.67150    0.21434 -17.130  < 2e-16 ***
## poly(x1, 3)3   -5.19304    0.21314 -24.364  < 2e-16 ***
## x4              0.98181    0.05919  16.588  < 2e-16 ***
## x5              0.15001    0.06015   2.494   0.0139 *
## x9             -1.93657    0.05604 -34.560  < 2e-16 ***
## x10            -0.06131    0.05650  -1.085   0.2799
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.194 on 128 degrees of freedom
## Multiple R-squared:  0.9545, Adjusted R-squared:  0.952
## F-statistic: 383.8 on 7 and 128 DF,  p-value: < 2.2e-16
```

```
anova(res)
```

```
## Analysis of Variance Table
##
## Response: y
##               Df Sum Sq Mean Sq   F value  Pr(>F)
## poly(x1, 3)    3 43.468  14.489  384.7957 < 2e-16 ***
## x4             1 10.955  10.955  290.9401 < 2e-16 ***
## x5             1  0.111   0.111    2.9401 0.08883 .
## x9             1 46.596  46.596 1237.4513 < 2e-16 ***
## x10            1  0.044   0.044    1.1778 0.27985
## Residuals    128  4.820   0.038
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now, after we tried to change the formula, what we can see now, the value of x10 became not significant at all and the value of x5 became less significant.

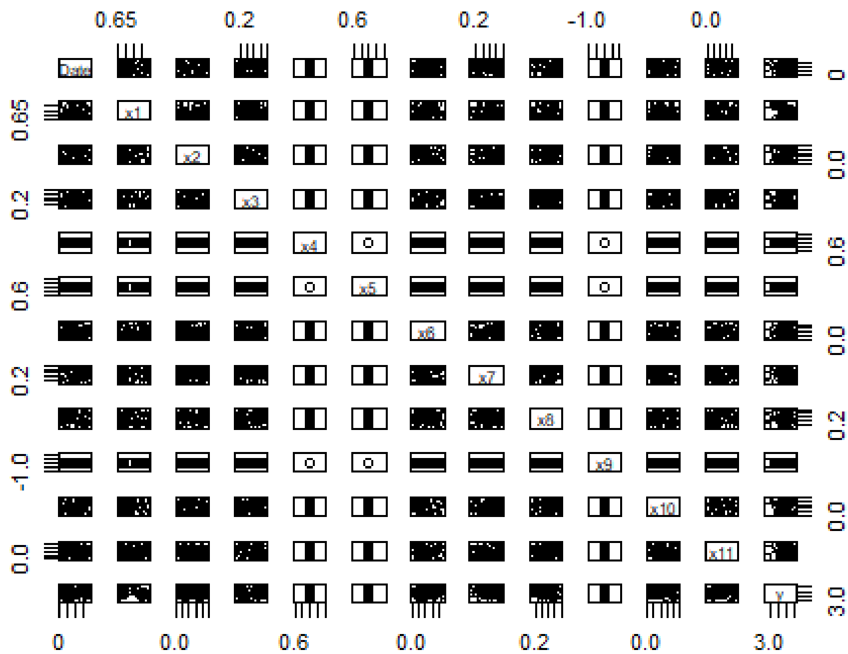
Now, let's try to create a new dataset again with parameter as below - x1= between 0.65 and 0.85 (for sampling) - x4=1 - x5=1 - x9=0 using the code below (it still in c)

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int cr=40;
    int row = 11;
    for(int i=0;i<cr;i++){
        for(int j=0;j<row; j++){
            if(j == 0){
                float value = (rand()%20)/100.0+0.65;
                printf("%.2f",value);
            }
            else if(j==3 || j==4)
                printf("1");
            else if(j==8)
                printf("0");
            else{
                float value = (rand()%100)/100.0;
                printf("%.2f",value);
            }
            if(j+1< row) printf(",");
        }
```

```
        puts("");
    }

    return 0;
}
```

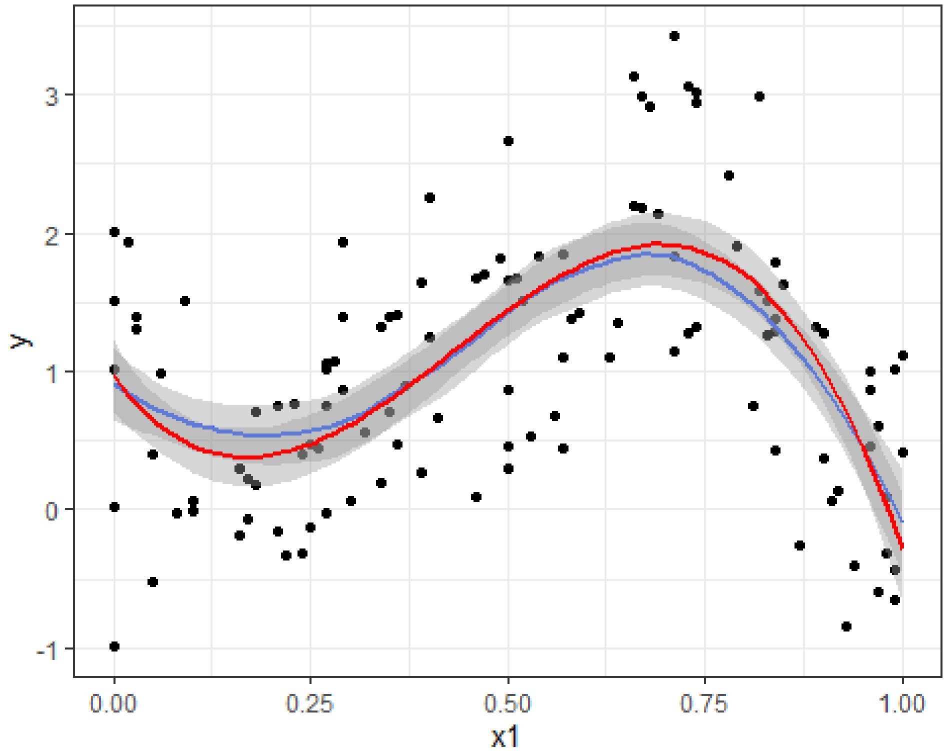
let's view the new dataset

```
d3 <- read.csv("20240201_0109_user_e7268-DoEShinnyApplication.csv")
d3 <- d3[137:176,]
summary(d3)

##      Date                 x1               x2               x3

##  Length:40          Min.   :0.6500   Min.   :0.0100   
Min.   :0.0500
##  Class :character   1st Qu.:0.6700   1st Qu.:0.2250   1st 
Qu.:0.2700
##  Mode  :character   Median :0.7150   Median :0.5150   
Median :0.5900
##                     Mean   :0.7265   Mean   :0.4943   
Mean   :0.5337
##                     3rd Qu.:0.7900   3rd Qu.:0.7475   3rd 
Qu.:0.7450
##                     Max.   :0.8400   Max.   :0.9900   
Max.   :0.9900
##        x4          x5          x6               x7               x8

##  Min.   :1   Min.   :1   Min.   :0.0100   Min.   :0.0500   
Min.   :0.040
##  1st Qu.:1   1st Qu.:1   1st Qu.:0.2375   1st Qu.:0.3350   1st 
Qu.:0.350
##  Median :1   Median :1   Median :0.3800   Median :0.6050   
Median :0.580
##  Mean   :1   Mean   :1   Mean   :0.4140   Mean   :0.5693   
Mean   :0.547
##  3rd Qu.:1   3rd Qu.:1   3rd Qu.:0.6075   3rd Qu.:0.8725   3rd 
Qu.:0.710
##  Max.   :1   Max.   :1   Max.   :0.9400   Max.   :0.9700   
Max.   :0.990
##        x9          x10              x11               y
##  Min.   :0   Min.   :0.000   Min.   :0.0000   Min.   :2.888
##  1st Qu.:0   1st Qu.:0.290   1st Qu.:0.2475   1st Qu.:3.293
##  Median :0   Median :0.435   Median :0.3850   Median :3.362
##  Mean   :0   Mean   :0.507   Mean   :0.4537   Mean   :3.359
##  3rd Qu.:0   3rd Qu.:0.765   3rd Qu.:0.7350   3rd Qu.:3.445
##  Max.   :0   Max.   :0.990   Max.   :0.9600   Max.   :3.653

plot(d3)
```



Let’s plot again:

```
ggplot(df, aes(y=y, x=x1)) + geom_point() + geom_smooth() +
geom_smooth(method = "lm", formula = y ~ poly(x,3), color="red") +
theme_bw()

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
d3[d3$y==max(d3$y),]
```

```
##                    Date   x1   x2   x3 x4 x5   x6   x7   x8 x9  x10
x11
## 150 2024-02-01-00:07:51 0.72 0.01 0.97  1  1 0.02 0.17 0.92  0 0.52
0.56
##            y
## 150 3.652516
```

Anyway, the optimal configuration is thus - x1=0.72 - x4=1 - x5=1 - x9=0. All other parameters are of no importance and the optimal value for y is around 3.65.